Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (previously presented): A method for persisting and recovering security keys in order to authorize a daemon or a command-line interface ("CLI"), comprising:

reading, with root as an effective user id, one or more security keys into a cache, wherein the root as effective user id enables the reading of root-only readable files including the one or more security keys and the one or more security keys can be accessed by a non-root user;

attempting to retrieve a private key from the cache using a real user id, wherein the cached certain security keys may include the private key and the private key may be used to digitally sign a message; and

determining if the private key was retrieved from the cache, wherein a failure to retrieve the private key from the cache indicates that authorization failed.

Claim 2 (previously presented): The method of claim 1, further comprising:

setting, with the root as the effective user id, the certain security keys, wherein the setting the certain security keys triggers performance of the reading one or more security keys into a cache.

Claim 3 (previously presented): The method of claim 2, wherein the setting the certain security keys comprises calling a setKeys method, wherein the setKeys method includes the reading one or more security keys into a cache.

Claim 4 (original): The method of claim 3, wherein a failure to retrieve the private key from the cache is caused by an error in the setKeys method.

Claim 5 (original): The method of claim 1, further comprising:

entering the CLI, wherein the CLI is entered by a non-root user on a managed node and the private key is a security key of the managed node.

Claim 6 (previously presented): A method for persisting and recovering security keys in order to authorize a daemon or a command-line interface ("CLI"), comprising:

reading, with root as an effective user id, one or more security keys into a cache, wherein the root enables the reading of files including the one or more security keys;

attempting to retrieve a private key from the cache using a real user id, wherein the cached certain security keys may include the private key and the private key may be used to digitally sign a message;

determining if the private key was retrieved from the cache, wherein a failure to retrieve the private key from the cache indicates that authorization failed;

entering the CLI, wherein the CLI is entered by a non-root user on a managed node and the private key is a security key of the managed node, wherein the managed node has a public key;

if the private key was retrieved from the cache, sending a message and a message copy, wherein the message copy is digitally signed with the private key from the managed node;

digitally signing the message with the managed node's public key;

comparing the message signed with the public key to the message copy signed with the private key; and

determining if the message is authorized based on the comparison of the message signed with the public key to the message copy signed with the private key.

Claim 7 (original): The method of claim 6, wherein the message comprises an executable, the method further comprising:

if the message is authorized, executing the executable.

Claim 8 (original): The method of claim 1, further comprising:

running a daemon process, wherein the daemon is run on a managed node and the private key is a security key of the managed node.

Claim 9 (previously presented): A method for persisting and recovering security keys in order to authorize a daemon or a command-line interface ("CLI"), comprising:

reading, with root as an effective user id, one or more security keys into a cache, wherein the root enables the reading of files including the one or more security keys;

attempting to retrieve a private key from the cache using a real user id, wherein the cached certain security keys may include the private key and the private key may be used to digitally sign a message;

determining if the private key was retrieved from the cache, wherein a failure to retrieve the private key from the cache indicates that authorization failed;

running a daemon process, wherein the daemon is run on a managed node and the private key is a security key of the managed node, wherein the managed node has a public key;

if the private key was retrieved from the cache, sending a message and a message copy, wherein the message copy is digitally signed with the private key from the managed node;

digitally signing the message with the managed node's public key;

comparing the message signed with the public key to the message copy signed with the private key; and

determining if the message is authorized based on the comparison of the message signed with the public key to the message copy signed with the private key.

Claim 10 (previously presented): The method of claim 1, wherein the reading one or more security keys into a cache is performed by an authentication class.

Claim 11 (original): The method of claim 10, wherein the cache is a private variable in the authentication class.

Claim 12 (previously presented): The method of claim 1, further comprising:

generating a security key pair, wherein the security key pair comprises the private key and a corresponding public key;

serializing the security key pair as a key file; and

storing the key file, wherein the reading one or more security keys into a cache comprises de-serializing the key file and reading the private key into the cache.

Claim 13 (currently amended): A tangible computer readable medium containing instructions for controlling a computer system to persist and recover security keys in order to authorize a daemon or a CLI, by:

reading, with root as an effective user id, one or more security keys into a cache, wherein the root as effective user id enables the reading of root-only readable files including the security keys and the one or more security keys can be accessed by a non-root user;

attempting to retrieve a private key from the cache using a real user id, wherein the cached one or more security keys may include the private key and the private key may be used to digitally sign a message; and

determining if the private key was retrieved from the cache, wherein a failure to retrieve the private key from the cache indicates that authorization failed.

Claim 14 (currently amended): The tangible computer readable medium of claim 13, further containing instructions for controlling the computer system by:

setting, with the root as an effective user id, the security keys, wherein the setting the security keys triggers the reading one or more security keys into a cache.

Claim 15 (currently amended): The tangible computer readable medium of claim 14, wherein the setting the security keys comprises calling a setKeys method, wherein the setKeys method includes the reading one or more security keys into a cache.

Claim 16 (currently amended): A tangible computer readable medium containing instructions for controlling a computer system to persist and recover security keys in order to authorize a daemon or a CLI, by:

reading, with root as an effective user id, one or more security keys into a cache, wherein the root enables the reading of files including the security keys;

attempting to retrieve a private key from the cache using a real user id, wherein the cached one or more security keys may include the private key and the private key may be used to digitally sign a message;

determining if the private key was retrieved from the cache, wherein a failure to retrieve the private key from the cache indicates that authorization failed, wherein the computer system comprises a managed node and the managed node has a public key;

if the private key was retrieved from the cache, sending a message and a message copy, wherein the message copy is digitally signed with the private key from the managed node;

digitally signing the message with the managed node's public key;

comparing the message signed with the public key to the message copy signed with the private key; and

determining if the message is authorized based on the comparison of the message signed with the public key to the message copy signed with the private key.

Claim 17 (previously presented): A method for persisting and recovering security keys in order to authorize a daemon or a CLI, comprising:

initializing an authentication class, wherein the authentication class comprises a setKeys method that includes a reading step;

calling, with root as an effective user id, the setKeys method of the authentication class, wherein the root as effective user id enables the reading of root-only readable files including security keys and the security keys can be accessed by a non-root user;

reading necessary security keys into a cache with the root; and

retrieving the necessary security keys from the cache using a real user id.

Claim 18 (original): The method of claim 17, wherein the cache is a private variable of the authentication class.

Claim 19 (original): The method of claim 17, wherein the necessary security keys are a private key of a managed node on which the authentication class is running and a public key of a central management server to which the managed node is operatively connected.

Claim 20 (original): The method of claim 17, wherein the authentication class is a Java class running in a Java Virtual Machine, the method further comprising:

initializing the Java Virtual Machine.

Claim 21 (previously presented): The method of claim 20 wherein the reading necessary security keys into the cache with the root enables the retrieving the necessary security keys from the cache using the real user id so long as the Java Virtual Machine is running.

Claim 22 (previously presented): The method of claim 1 wherein the reading one or more security keys into the cache with root as the effective id is performed at startup of a Java Virtual Machine and enables the retrieval of the one or more security keys from the cache using a real user id so long as the Java Virtual Machine is running.

Claim 23 (currently amended): The tangible computer readable medium of claim 13 wherein the reading one or more security keys into the cache with root as the effective id is performed at startup of a Java Virtual Machine and enables the retrieval of the one or more security keys from the cache using a real user id so long as the Java Virtual Machine is running.